INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
*(Volume 8, Issue 2 - V8I2-1282)*
Available online at: https://www.ijariit.com

# Microservices: Architecture, containers, and challenges

*Rohan Talekar*
*talekarrohan01@gmail.com*
*Patkar Varde College, Mumbai, Maharashtra*

## ABSTRACT

***Microservices (also known as microservices architecture) is a cloud-native architectural paradigm in which a single application is built of several loosely connected and independently deployable, tested, or developed. The microservice architecture concept says, that you break a big application into small bodies into small services of loosely coupled applications of services. This paper discusses monolithic architecture, service-oriented architecture and microservice architecture, and then container technology. Then move on to microservices challenges.***

***Keywords***: *Architecture, Monolithic Architecture, Service-Oriented Architecture (SOA), Microservice Architecture, Container, Challenges Of Microservices.*

## 1. INTRODUCTION

Microservices have been one of the most popular architectural concepts in software engineering in recent years, with an increasing number of cloud computing applications adopting them. Most of the companies are using microservices like Netflix, Amazon, eBay, Sound Cloud, Uber, etc.

The monolithic architecture resembles a large container in which all of an application's software components are built and firmly packed. The advantage of monolithic architecture is very simple to develop and deploy. Also, you can scale your application. Microservices, also known as Microservices Architecture, is an architectural style in which an application is structured as a collection of tiny independent services based on a business domain. Each service in a microservices architecture is self-contained and implements only one business feature. Microservices are characterized by their independence in development, deployment, and release, as well as low coupling. Container technology, sometimes known as merely a container, is a means of packaging a program such that it may be executed independently of other processes, with all of its dependencies.

## 2. ARCHITECTURES

There is three architecture we used in applications this are monolithic architecture, service-oriented architecture, and microservice architecture.

### A. Monolithic Architecture

Monolithic means “mono” means single and “lithic” means stone.

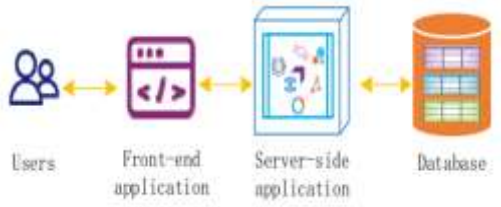


**Figure 1: Monolithic Architecture**

A monolithic application is the one giant box or one giant container in which the whole application system resides. This was predominantly the most used architecture and it is still one of the most heavily used architecture because of this very straightaway advantage, it is very easy to deploy. The complexity is very low. so, you just simply need to get an instance of a server, simply set up your application deploy or application set up your underlying database and you are good to go. In Monolithic Architecture, Any request going to this application from a user machine via HTTP request goes directly to that application server, and then that application server has an underline database with which it interacts and sends back the data and the request which client or user has requested for. [1] [2] [3]

### B. Service-Oriented Architecture:

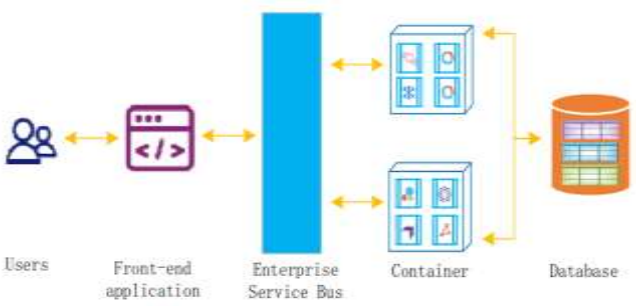


**Figure 2: Service Oriented Architecture.**

 

SOA is an architectural approach for developing software applications that take advantage of network services such as the web. In SOA, we created independent services. There are different services it can be reused as the same services and everything is loosely coupled. Forex. A bunch of code is service 1, a bunch of code is service 2, and a bunch of code is service 3 so, they don't need to understand the entire flow they can work independent services. Different containers can be used to handle each service. All the services (communicate, share) in this use the same database. Independent small services in the SOA make it easy to test and debug applications rather than troubleshooting enormous code blocks, making the service-oriented architecture very trustworthy. [1] [2] [3]

### C. Microservice Architecture


**Figure 3: Microservice Architecture.**

Microservice architecture breaks its application into small pieces into small services that are very loosely coupled .so, all those services interact with each other all these services are a kind of autonomous body.

The microservice architecture concept says, that you break a big application into small bodies into small services of loosely coupled applications of services.

In Microservice Architecture, All these services are working independently because all these are different microservices. So, it can be MS1, MS2, MS3, MS4, and MS5 so these different microservices can be deployed separately and can be accessed through API Gateway. [1] [2] [3]

API Gateway means, that whenever the client sends a request it will directly go to an API Gateway which acts as an entry point. This entry points understand what is exact request the user is sending and based on the service which is requested it identifies which particular API and service can cater to that particular request and it goes to that particular service take that input and send it back to the user. API Gateway stands as an interface between the user and a client machine and is a loosely coupled service. Also, all the different services have their independent database because all these business processes are having their specific database and their data model and this helps because there is no reliance on any other services that's why it is it can work independently.

## 3. CONTAINERS

It is a one-line definition is Isolated Packaged Environment.

Another definition is "An object for holding and transporting something" which means you can hold something here and you can transport it anywhere else transporting something is a very powerful feature of a container that means containers are very portable in the sense that you can run containers on your on-premise data centers on your laptop or any of the cloud providers like AWS, Microsoft Azure, GCP.

Isolated Package Environment so, it says that it's a package environment that is isolated. So, by package environment, it means that it contains the complete package for an application.

The package would be for an application it would contain the operating system itself then the application dependencies for example if you are running a python web application you should have a flask in it and then of course you should have python in it and then the application itself now if you relate these two or if you combine these two isolated packaged environment which is portable or which can be transported that means you can run the application anywhere it can be your laptop or on any of AWS, Microsoft Azure, and Google Cloud Platform are examples of cloud providers. [4] [5]

### A. Evolution Of Container

So, in the Initial days or the legacy day applications used to be hosted in different ways.

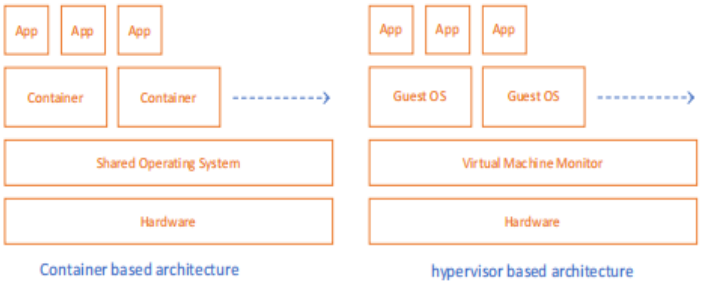


**Figure 4: container- and hypervisor-based virtualization architectures.**

We used to have hardware CPU, RAM, Storage networks, firewalls, etc. then we used to install an operating system on top of the hardware like Linux, windows, or mac, and then we used to install our application or run our application or host our application on this application. You could host multiple applications on the same operating system which again typically was never suggested but if you wanted you could. So, it is a server environment is your server installs an operating system on your server and then you host your application on top of the server. Single server to a single application or multiple applications. [1] [4] [5]

Then things were changed when virtualization was introduced the reason virtualization become so famous was because in a single environment or basically in the server environment or in this a lot of resources were being wasted. So, virtualization breaks that gap it utilizes the resources in a very nice way and reduces cost. So, you have the same hardware CPU, RAM, Storage network, etc. Instead of your operating system, you have a hypervisor here hypervisor like VMWare, zen, hyper-v. zen is used by AWS and hyper-v is used by Microsoft Azure and the hypervisor runs multiple virtual machines. You can host one or more multiple applications on a single virtual machine. The operating system that operates within a virtual machine is known as a guest operating system. For virtual machines produced using vApp templates, you may install a guest os and control guest os customization. [1] [4] [5]

### B. Then comes to the containers

 

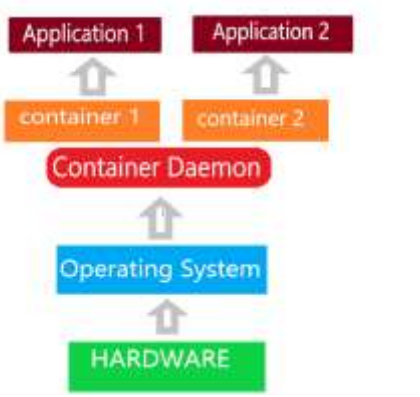


**Figure 5: Container.**

Underlying infrastructure remains the same nothing change you have we still have our hardware basically CPU, RAM, Storage network, etc., and then we have our operating system on top of the hardware or operating system on top of the VM also so, basically, this operating system could as well have been on a virtual machine but now we have something called containers daemon or the things which run containers. Container technology using container daemon we run containers and each container would house a single application remember this you cannot have multiple application on a single container you can have only one application in one container and each container are isolated from another container that means container one would never know that container two is running and each of them is a packaged environment that means container one would contain an operating system on an application environment which can be different from container two. Which might contain a different application in a different operating system. [1] [4] [5]

For example. Container 1 can contain a java application, container 2 can contain a python application running on the same operating system and using the same hardware now this operating system can as I said to be hosted on top of a virtual machine as well.

## 4. CHALLENGES IN MICROSERVICES

A. Apps don't scale with storage, and performance is unreliable.

B. It is extremely difficult to securely transfer data across cities and/or cloud service providers.

C. **Testing:** Microservices-based systems make the testing part of every software development lifecycle (SDLC) more difficult. Because each microservice is self-contained.

D. **Design:** It's critical to plan for failure. You must be prepared to deal with a variety of failures, including system outages, delayed service, and unexpected reactions.

## 5. CONCLUSION

This paper discussed three architectures monolithic architecture, service-oriented architecture, and microservice architecture. When picking a microservices architecture, keep the needs of the company in mind. In most cases, there isn't a single optimum development model. Load balancing is important for delayed services. For testing, microservices it's necessary to test each one separately. Your microservices design should be able to withstand a decent amount of failure. Resiliency tests and fault detection are therefore essential. Many coding operations are automated, drastically lowering development time and expenses. The cloud functionality option lowers operational expenses and improves performance.

## 6. ACKNOWLEDGEMENT

I'd like to express my heartfelt gratitude to my teacher, Ms. Manali Patil, for providing me with the wonderful opportunity to write this wonderful research paper on the topic “Microservices: Architecture, containers and Challenges”, which also allowed me to do a lot of research and learn a lot of new things. I'm grateful to them.

## 7. REFERENCES

**[1]** Guozhi Liu, Bi Huang, Zhihong Liang, Minmin Qin, Hua Zhou, Zhang Li “Microservices: architecture, container, and Challenges”. Published in: 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C). Date of Conference**:** 11-14 Dec. 2020.

**[2]** Omar Al-Debagy, Peter Martinek “A Comparative Review of Microservices and Monolithic Architectures” Published in: 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI). Date of Conference: 21-22 Nov. 2018.

**[3]** Mohammad Sadegh Hamzehloui, Shamsul Sahibuddin, and Ardavan Ashabi “A Study on the Most Prominent Areas of Research in Microservices”. International Journal of Machine Learning and Computing, Vol. 9, No. 2, April 2019.

**[4]** Tamanna Siddiqui, Shadab Alam Siddiqui, Najeeb Ahmad Khan “Comprehensive Analysis of Container Technology” s and Computer Networks (ISCON) GLA University, Mathura, UP, India. Nov 21-22, 2019.

**[5]** Vitor Goncalves da Silva, Marite Kirikova, Gundars Alksnis “Containers for Virtualization: An Overview” ISSN 2255-8691 (online) ISSN 2255-8683 (print) May 2018, vol. 23, no. 1, pp. 21–27.

 